The Impact of Morphware on Parallel Computing

Reiner Hartenstein
Kaiserslautern University of Technology
The White House, Sept 2000: Bill Clinton condemns the Digital Divide in America

World Economic Forum 2002: the Global Digital Divide, disparity between the "haves" and "have nots"

Another digital divide, not only in America:

Parallel Computing / High Performance Computing: distinguishing between the "have nots" and "haves", having qualifications also for morphware, configware and embedded system design
Preface (2): Why Embedded Systems?

The real labor market: 10 times more programmers will write embedded applications than computer software by 2010.
• Introduction
• Reconfigurable Computing
• Flowware
• Datastream-based Computing
• The Anti Machine Paradigm
• Final Remarks
Silicon Application Revolutions

- 1957: TTL
- 1967: LSI, MSI
- 1977: μproc., memory
- 1987: ASICs, accel's
- 1997: reconfigurable
- 2007: morphware: fastest growing sector of the semiconductor market
Makimoto’s Wave

**Tredennick’s Paradigm Shifts**
- **standard**
  - algorithm: fixed
  - resources: fixed
- **hardwired**
  - algorithm: fixed
  - resources: fixed
- **procedural programming**
  - algorithm: variable
  - resources: fixed
- **structural programming**
  - algorithm: variable
  - resources: variable

**vN machine paradigm**

**Coarse grain RAs**

**Hartenstein’s Curve**

**no further wave!**

© 2004, reiner@hartenstein.de

http://hartenstein.de
The Impact of Makimoto's Paradigm Shifts

Software Industry's Secret of Success

Repeat Success Story by new Machine Paradigm!

Personalization (CAD) before fabrication

Procedural personalization via RAM-based Machine Paradigm

structural personalization: RAM-based before run time

Dr. Makimoto: FPL 2000 keynote

Configware

Software Industry's Secret of Success

Repeat Success Story by new Machine Paradigm!

Personalization (CAD) before fabrication

Procedural personalization via RAM-based Machine Paradigm

structural personalization: RAM-based before run time

Dr. Makimoto: FPL 2000 keynote

Configware
Limitations of traditional Parallelism

- the Havenots
- only instruction-stream-based Parallelism

© 2004, reiner@hartenstein.de
http://hartenstein.de

Kaiserslautern University of Technology
Problems with the von Neumann paradigm

vN: unbalanced
vN bottleneck

CPU

stolen from Bob Colwell
http://laughingriot.incide.net
“Pollack’s Law” (simplified)

- Growth factor
- Area efficiency
- Performance

Legend:
- [intel]
more and more efforts yield only marginal improvements

areas fade away

dataflow machines dead

shrinking supercomputing conferences

98.5% vN-only

this monopoly is the problem

11 ISCA papers 1973-2001

[David Padua, John Hennessy]
.. but the von Neumann Paradigm is still the dominant doctrine ...

... still pushing he basic models from the times of mainframe dinosaurs

Microelectronics is ignored (except falling cost of computational effort)

A Re-orientation is over-due

after >10 technology generations ...

• 1\textsuperscript{st} 4004 ...
• 2\textsuperscript{nd} 8008 ...
• 3\textsuperscript{rd} 8086 ...
• 4\textsuperscript{th} 80286 ...
• 5\textsuperscript{th} 80386 ...
• 6\textsuperscript{th} 80486 ...
• 7\textsuperscript{th} P5 (Pentium) ...
• 8\textsuperscript{th} P6 (Pentium Pro / Pentium II) ...
• 9\textsuperscript{th} Pentium III ...
• 10\textsuperscript{th} .... ...
• 11\textsuperscript{th} ....

... the vN Microprocessor is a methusela, the steam engine of the silicon age.

computed sciences are ultra conservative ...

... to avoid saying: senile
The Digital Divide of Parallelism

many application areas: most processors idling

the have nots

only instruction-stream-based Parallelism

the haves

have also data-stream-based Parallelism

less watts per MIPS by orders of magnitude

parallelism also instruction-level (ILP) and below

Dead Supercomputer Society

[Keynote by Gordon Bell at ISCA 2000]

- ACRI
- Alliant
- American Supercomputer
- Ametek
- Applied Dynamics
- Astronautics
- BBN
- CDC
- Convex
- Cray Computer
- Cray Research
- Culler-Harris
- Culler Scientific
- Cydrome
- Dana/Ardent/ Stellar/Stardent
- DAPP
- Denelcor
- Elexsi
- ETA Systems
- Evans and Sutherland
- Computer
- Floating Point Systems
- Galaxy YH-1
- Goodyear Aerospace MPP
- Gould NPL
- Guiltech
- ICL
- Intel Scientific Computers
- International Parallel Machines
- Kendall Square Research
- Key Computer Laboratories
- MasPar
- Meiko
- Multiflow
- Myrias
- Numerix
- Prisma
- Tera
- Thinking Machines
- Saxpy
- Scientific Computer Systems (SCS)
- Soviet Supercomputers
- Supertek
- Supercomputer Systems
- Suprenum
- Vitesse Electronics
Lacking Sense of Direction?

„we are o.k. !“ (no new direction)

blinders:
for ignoring the impact of Morphware

© 2004, reiner@hartenstein.de

http://hartenstein.de
• Introduction
• Reconfigurable Computing
• Flowware
• Datastream-based Computing
• The Anti Machine Paradigm
• Final Remarks
IT ages

mainframe age

computer age (PC age)

morphware age

1957

1967

1977

1987

1997

2007

© 2004, reiner@hartenstein.de

http://hartenstein.de
Steroids for the aging microprocessor:

Reconfigurable Computing
control-procedural vs. data-procedural

The structural domain is primarily data-stream-based:

Flowware

..... mostly not yet modelled that way:
most flowware is hidden by its indirect
instruction-stream-based implementation

Flowware provides a (data-)procedural abstraction
from the (data-stream-based) structural domain

Flowware converts „procedural vs. structural“
into „control-procedural vs. data-procedural“ ...

... a Troyan horse to introduce the structural domain
to the procedural-only mind set of programmers
focusing on coarse grain

• Fine Grain morphware platforms
  
  already mainstream: reconfigurable logic
  just logic design on a strange platform

• Coarse Grain platforms:
  Reconfigurable Computing:
  not that new – but shocking the fundamentals of CS curricula

speed-up til 3 orders of magnitude

an order of magnitude more MIPS/mW than fine grain
• Introduction
• Reconfigurable Computing
• Flowware
• Datastream-based Computing
• The Anti Machine Paradigm
• Final Remarks
IT ages

flowware data streams ...

mainframe age

computer age (PC age)

morphware age


IBM®

intel.

Microsoft®
Flowware defines:

... which data item at which time at which port

DPA

input data streams

output data streams
Mathematicians like to wax rhapsodic about the elegance, beauty and depth of their proofs. But computer proofs yield truth without understanding.

John Horgan: "The End of Science Revisited"

Mathematicians liked the beauty of Systolic Arrays. Due to a lack of depth in understanding systolic computing their efforts yielded only poor synthesis algorithms.

Reiner Hartenstein
computing paradigms and methodologies

1946: machine paradigm (von Neumann)
1980: data streams (Kung, Leiserson)
1989: anti machine paradigm
1990: rDPU (Rabaey)
1994: anti machine high level programming language
1995: super systolic array (rDPA)
1996+: SCCC (LANL), SCORE, ASPRC, Bee (UCB), ...
1997+: discipline of distributed memory architecture
1997+: configware / software partitioning compiler
Supersystolic Array Principles

- take systolic array principles
- replace classical synthesis by simulated annealing
- yields the supersystolic array
- a generalization of the systolic array
- no more restricted to regular data dependencies
- now reconfigurability makes sense: use morphware
• Data-stream-based Computing is heading for mainstream

- 1997  SCCC (LANL) Streams-C Configurable Computing
- SCORE (UCB) Stream Computations Organized for Reconfigurable Execution
- ASPRC (UCB) Adapting Software Pipelining for Reconfigurable Computing
- 2000 Bee (UCB), ...

- Most stream-based multimedia systems, etc.
- Many other areas ....

Flowware:
managing data streams
Software:
managing instruction streams
Flowware Languages

specialized:

**Brook**: for modern graphics hardware

**Streams-C**: defines 1-D streams; generates VHDL

**DSP-C**: allows to describe key features of DSPs

general purpose:

**MoPL**: fully supporting the anti machine paradigm
- the counterpart of the von Neumann paradigm
### Programming Language Paradigms

<table>
<thead>
<tr>
<th>Language Category</th>
<th>Von Neumann Languages</th>
<th>Anti Machine Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both deterministic</td>
<td>procedural sequencing: traceable, checkpointable</td>
<td>read next data item, goto (data addr.), jump (to data addr.), data loop, loop nesting, parallel loops, escapes, data stream branching</td>
</tr>
<tr>
<td>Operation sequence driven by:</td>
<td>read next instruction, goto (instr. addr.), jump (to instr. addr.), instr. loop, loop nesting no parallel loops, escapes, instruction stream branching</td>
<td>read next data item, goto (data addr.), jump (to data addr.), data loop, loop nesting, parallel loops, escapes, data stream branching</td>
</tr>
<tr>
<td>State register</td>
<td>program counter</td>
<td>data counter(s)</td>
</tr>
<tr>
<td>Address computation</td>
<td>massive memory cycle overhead</td>
<td>overhead avoided</td>
</tr>
<tr>
<td>Instruction fetch</td>
<td>memory cycle overhead</td>
<td>overhead avoided</td>
</tr>
<tr>
<td>Parallel memory bank access</td>
<td>interleaving only</td>
<td>no restrictions</td>
</tr>
<tr>
<td>Language features</td>
<td>control flow + data manipulation</td>
<td>data streams only (no data manipulation)</td>
</tr>
</tbody>
</table>

- **very easy to learn**
- **multiple GAGs**
- **much more powerful**
- **much more simple**

© 2004, reiner@hartenstein.de
• Introduction
• Reconfigurable Computing
• Flowware
• Datastream-based Computing
• The Anti Machine Paradigm
• Final Remarks
Software vs Flowware and Configware

Programming source for instruction-stream-based computing (von Neumann etc.):

Software

The programming source for data-stream-based computing (the anti machine paradigm):

Flowware

Programming sources for Reconfigurable Computing (morphware):

Flowware and Configware

Sources for Embedded Systems:

Flowware, Configware and Software

© 2004, reiner@hartenstein.de
http://hartenstein.de
Paradigm Shifts: Nick Tredennick's view

Instruction-stream-based computing:
- Algorithms variable
- Resources fixed

Data-stream-based reconfigurable computing:
- Algorithms variable
- Resources variable

Software
- 3 sources

Configware
- Programmable

Flowware
- μprocessor
- Accelerators
Importance of binding time

Not all switching is done by Configware

Configuration is like a kind of pre-packed frozen-in "super instruction fetch"

But you can't implement decisions!

structural domain

generate datapath

Fabrication time

But you can't implement decisions!

© 2004, reiner@hartenstein.de

http://hartenstein.de
Machine paradigms

von Neumann instruction stream machine

(reconf.) data-stream machine (anti machine)

Flowware

Software

distributed memory architecture*

*) the new discipline came just in time: see Herz et al.: Proc. IEEE ICECS, 2002
also see books by Francky Catthoor et al.
Configware / Flowware Compilation

**asM**

**rDPA**

**r. Data Path Array**

**Distributed Memory Architecture**

**Data Streams**

**Wrapper**

**Intermediate**

**Mapper**

**Configware**

**Flowware**

**Scheduler**

**Data Sequencer**

**Address Generator**

"Instruction" fetch before runtime
mapping algorithms efficiently onto rDPA

SNN filter on KressArray

10 ports

array size: 10 x 16 = 160 rDPUs

rout thru only

backbus connect

not used

© 2004, reiner@hartenstein.de

http://hartenstein.de
PACT XPP: Reference Module: XPU128 Co-Processor

XPP128 rDPA

- Full 32 or 24 Bit Design *working silicon*
- 2 Configuration Hierarchies
- Evaluation Board available, and
- XDS Development Tool with Simulator

© PACT AG, Munich
http://pactcorp.com
The Anti Machine Paradigm

- Introduction
- Reconfigurable Computing
- Flowware
- Datastream-based Computing
- The Anti Machine Paradigm
- Final Remarks
**IT ages**

**mainframe age**

**computer age (PC age)**

**morphware age**

von Neumann does not support morphware

flowware  data streams ...

© 2004, reiner@hartenstein.de

http://hartenstein.de
flowware  data streams ...

mainframe age

computer age (PC age)

morphware age

PC replaced by PS

(personal supercomputer)

IBM

Intel

Microsoft

co-compiler

µProc.
rDPA


© 2004, reiner@hartenstein.de

http://hartenstein.de
entire system on a single chip
all you need on board

- Xilinx Virtex-II Pro FPGA Architecture
- PowerPC 405 RISC CPU (PPC405) cores
- FPGA Fabric-based on Virtex-II Architecture

Source: Ivo Bolsens, Xilinx
We need 2 machine paradigms

Tredennick's Paradigm Shifts

- **1957**: hardwired algorithm: fixed, resources: fixed
- **1967**: procedural programming algorithm: variable, resources: fixed
- **1977**: structural programming algorithm: variable, resources: variable
- **2007**: vN machine paradigm anti machine paradigm

Coarse grain RAs
Why a dichotomy of machine paradigms?

vN: unbalanced

The anti machine has no von Neumann bottleneck

vN bottleneck

data stream machine:
  • bad message: caches do not help
  • good message: no vN bottleneck
  • caches not needed

stolen from Bob Colwell
http://laughingriot.incide.net
High level PL source

“vN” machine paradigm

Partitioner

SW compiler
Analyzer / Profiler
CW compiler

SW code
CW Code

Resource Parameters

could provide the platforms
supporting different platforms

© 2004, reiner@hartenstein.de
• Introduction
• Reconfigurable Computing
• Flowware
• Datastream-based Computing
• The Anti Machine Paradigm
• Final Remarks
Flowware heading toward mainstream

- Data-stream-based Computing is heading for mainstream

  - 1997 SCCC (LANL) Streams-C Configurable Computing
  - SCORE (UCB) Stream Computations Organized for Reconfigurable Execution
  - ASPRC (UCB) Adapting Software Pipelining for Reconfigurable Computing
  - 2000 Bee (UCB), ...
  - Most stream-based multimedia systems, etc.
  - Many other areas ....

Flowware ...... mostly not yet modelled that way: most flowware is hidden by its indirect instruction-stream-based implementation

© 2004, reiner@hartenstein.de
http://hartenstein.de
All enabling technologies are available

- literature from last 30 years
- languages & (co-)compilation techniques
- anti machine and all its architectural resources
- parallel memory IP cores and generators
- morphware vendors like PACT ....
- anything else needed
What's the problem?

The Digital Divide of CS

It's the gap between **procedural** (instruction-stream-based) and **structural** (datastream-based) mind set

Traditional CS: programming is (control-)procedural-only,

The von Neumann paradigm's monololy is the main problem

We urgently need an upgrade of CS & CSE curricula to avoid a qualification disaster on the IT job market

The typical programmer has problems to understand function evaluation without machine mechanisms....

The anti machine as the 2\textsuperscript{nd} paradigm is the key to cope with this problem - for successful curricular innovation
thank you
END
for discussion:

>>> for discussion <<<

© 2004, reiner@hartenstein.de
http://hartenstein.de

Kaiserslautern University of Technology

52
RWC Real World Computing, Japan, 40 TFLOPS

Crossbar weight: 220 t, 3000 km Kabel,
5120 Processors, 5000 pins each
Loop Transformation Examples

**Sequential Processes:**

- Loop 1-16
- Body
- Endloop

- Loop 1-8
- Body
- Body
- Endloop

- Loop 1-8
- Fork

- Loop 1-8
- Loop 9-16
- Body
- Body
- Endloop

- Loop 1-2
- Join

- Strip mining

**Resource Parameter Driven Co-Compilation**

**Host:**

- Loop 1-8
- Trigger
- Endloop

**reconf.array:**

- Loop 1-4
- Trigger
- Endloop

- Loop 1-8
- Trigger
- Endloop

- Loop 1-2
- Trigger
- Endloop
We introduce: Co-Compilation

Hardware / Software Co-Design turns to Configware / Software Co-Design

Software running on Computer Machine Paradigm

high level programming language source

partitioning compiler

Reconfigurable Accelerators

Reconfigurable Architecture (RA) -- instead of hardwired

Configware running on Xputer "Soft" Machine Paradigm

no CAD! Compilation instead!

© 2004, reiner@hartenstein.de

http://hartenstein.de
### Machine Paradigms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>driven by:</strong></td>
<td>Instruction streams</td>
<td>data streams (no “dataflow”)</td>
</tr>
<tr>
<td>engine principles</td>
<td>instruction sequencing</td>
<td>sequencing data streams</td>
</tr>
<tr>
<td>state register</td>
<td>single program counter</td>
<td>(multiple) data counter(s)</td>
</tr>
<tr>
<td>Communication path set-up (&quot;instruction fetch&quot;)</td>
<td>at run time</td>
<td>at load time</td>
</tr>
<tr>
<td>data path</td>
<td>DPU (e.g. single ALU)</td>
<td>DPU or DPA (DPU array) etc.</td>
</tr>
<tr>
<td>operation</td>
<td>sequential</td>
<td>parallel pipe network etc.</td>
</tr>
</tbody>
</table>

*) e.g. Bee project Prof. Broderson

© 2004, reiner@hartenstein.de
Xplorer Plot: SNN Filter Example

route-thru-only rDPU

2 hor. NNports, 32 bit
3 vert. NNports, 32 bit

result
operator
operand
route thru
backbus connect
• take systolic array principles
• replace classical synthesis by simulated annealing
• yields the super systolic array
• a generalization of the systolic array
• no more restricted to regular data dependencies
• now reconfigurability makes sense
### Super Pipe Networks

The key is *mapping*, rather than architecture.

<table>
<thead>
<tr>
<th>Array Type</th>
<th>Applications</th>
<th>Pipeline Properties</th>
<th>Mapping</th>
<th>Scheduling (Data Stream Formation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>systolic array</td>
<td>regular data dependencies only</td>
<td>linear only, uniform only</td>
<td>linear projection or algebraic synthesis</td>
<td></td>
</tr>
<tr>
<td>super-systolic rDPA</td>
<td>no restrictions</td>
<td></td>
<td>simulated annealing or P&amp;R algorithm</td>
<td>(e.g. force-directed) scheduling algorithm</td>
</tr>
</tbody>
</table>

*) KressArray [1995]
KressArray Family generic Fabrics: a few examples

Examples of 2nd Level Interconnect: laid out over rDPU cell - no separate routing areas!

Select mode, number, width of NNports

Select Function Repertory

rout-through and function

rout-through only

more NNports: rich Rout Resources

select Nearest Neighbour (NN) Interconnect: an example

http://kressarray.de

http://hartenstein.de
KressArray Xplorer

User Interface
Architecture Editor
Mapping Editor

ALEX Code
ALE-X Compiler
Architecture Estimator

interm. form
interm. form

HDL Generator Simulator
VHDL Verilog
Design Rules
Datapath Generator Generator
KressrDPU Layout

Power Estimator
Power Data

Mapper
Scheduler
DPSS
Analyzer

Bus & I/O Schedule
Delay Estim.

Suggestion
statist. Data

Inference Engine (FOX)

interm. form
interm. form

Application Set

KressArray (Design Space) Platform Space Explorer

new
Improvement Proposal Generator

http://kxplorer.informatik.uni-kl.de
The Secret of Success: Co-Compilation

supporting platform-based design

High level PL source

“vN” machine paradigm

Partitioner

SW compiler

Analyzer / Profiler

CW compiler

SW code

CW Code

supporting different platforms

Resource Parameters
Loop Transformation Examples

Sequential processes:

- Loop 1-16
- Body
- Endloop
- Loop 1-8
- Body
- Body
- Endloop
- Loop
- Unrolling
- Fork
- Join
- Strip mining

Resource parameter driven Co-Compilation

Host:
- Loop 1-8
- Trigger
- Endloop
- Loop 1-4
- Trigger
- Endloop
- Loop 1-2
- Trigger
- Endloop

Reconf. array:
- Loop 1-8
- Trigger
- Endloop
- Loop 1-4
- Trigger
- Endloop
- Loop 1-2
- Trigger
- Endloop
We introduce: Co-Compilation

Hardware / Software Co-Design turns to Configware / Software Co-Design

Software running on Computer Machine Paradigm

Configware running on Xputer "Soft" Machine Paradigm

Reconfigurable Accelerators

Reconfigurable Architecture (RA) -- instead of hardwired

High level programming language source

Partitioning compiler

no CAD! Compilation instead!
Significance of Address Generators

- Address generators have the potential to reduce computation time significantly.

- In a grid-based design rule check a speed-up of more than 2000 has been achieved, compared to a VAX-11/750.

- Dedicated address generators contributed a factor of 10 - avoiding memory cycles for address computation overhead.
• Embedded System Design Crisis
• Computing Crisis
• CS for Embedded Systems?
• Flowware-based Computing
• Enabling Architectural Resources
• New Machine Paradigm
• final remarks
Why a dichotomy of machine paradigms?

vN: unbalanced

vN bottleneck

data stream machine:
- bad message: caches do not help
- good message: no vN bottleneck
- caches not needed

The anti machine has no von Neumann bottleneck

© 2004, reiner@hartenstein.de
http://hartenstein.de
Kaiserslautern
University of Technology
67
Acceleration Mechanisms

- parallelism by multi bank memory architecture
- auxiliary hardware for address calculation
- address calculation before run time
- avoiding multiple accesses to the same data.
- avoiding memory cycles for address computation
- improve parallelism by storage scheme transformations
- improve parallelism by memory architecture transformations
- alleviate interconnect overhead (delay, power and area)
Synthesizable distributed memory architecture...
for a Stream-based Soft Machine

Scheduler

“instructions”

rDPA

Memory (data memory)
memory bank
memory bank
memory bank
memory bank

Sequencers (data stream generator)
Migration of programming to the structural domain

The structural domain has become RAM-based

The opportunity to introduce the structural domain to programmers ...

... to bridge the gap by clever abstraction mechanisms using a simple new machine paradigm
Very high throughput on low power slow FPGAs may be obtained only by algorithmic cleverness - not yet taught by CS & CSE at Universities - an urgent educational problem.
Summary of the Anti Machine Paradigm

- anti language primitives are almost the same (slightly extended)
- anti machine execution potential is dramatically more powerful
- provides drastically more flexibility
- not always replacing von Neumann
old CS lab course philosophy:
given an application: implement it by a program

new CS freshman lab course environment:
Given an application:
   a) implement it by writing a program
   b) implement it as a morphware prototype
   c) Partition it into P and Q
      c.1) implement P by software
      c.2) implement Q by morphware
      c.3) implement P / Q communication interface
Semiconductor Paradigm Shifts

“Mainstream Silicon Application is switching every 10 Years”

“The Programmable System-on-a-Chip is the next wave”

Makimoto's Wave

Tredennick's Paradigm Shifts

© 2004, reiner@hartenstein.de
http://hartenstein.de
There are exceptions

GI / ITG Fachgruppe PARS (Parallele Algorithmen, Rechnerstrukturen und Systemsoftware)
also turns its attention to Reconfigurable Computing (keynote at joint 19th PARS / 33rd speed-up workshop Basel, Switzerland, March 2003)

Andt Bode (general chair ISCA 2004, Munich) now also is interested in Reconfigurable Computing
Fig. 8. Increasing mask set cost and total NRE cost.
new kinds of concurrency are becoming important

chip-level multiprocessing + simultaneous multithreading

many bugs relate to concurrency issues

greatly complicates the verification process
Steroids for the aging microprocessor:
The Impact of Reconfigurable Computing
processor/memory communication bottleneck

vN: unbalanced

vN bottleneck

CPU

caches, ...

stolen from Bob Colwell
http://laughingriot.incide.net
Throughput vs. Flexibility

- **coarse grain** goes far beyond bridging the gap

- Throughput
- Flexibility

- **FPGAs**
- ** instruction set processors**
- **hardwired**
- **rDPAs (reconfigurable computing)**
- **standard microprocessor**
- **DSP**

[T. Claasen et al.: ISSCC 1999]
[*) R. Hartenstein: ISIS 1997
[Diagram (except *) by Hugo De Man]

© 2004, reiner@hartenstein.de
http://hartenstein.de
**Wide variety of speed-up factors**

**Key issue: algorithmic cleverness**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Application Example</th>
<th>Speed-up Factor</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACT Xtreme 4-by-4 array [2003]</td>
<td>16 tap FIR filter</td>
<td>x16 MOPS/mW</td>
<td>Straight forward</td>
</tr>
</tbody>
</table>
| MoM anti machine with DPLA* [1983] | grid-based DRC**  
1-metal 1-poly nMOS  
256 reference patterns | > x1000 (computation time) | Multiple aspects   |

*) MPC fabrication via E.I.S. multi university project  
**) Design Rule Check
Improving RTL-only design cost model

- tall thin engineer
- small block reuse
- large block reuse
- IC implementation tools
- Intelligent testbench
- ES level methodology

RTL methodology only
w. future improvements

mostly system level issues

Microprogramming to replace FSM design

Hardware languages replace EE-type schematics

EDA Software and its interfacing languages

Newer system level languages like systemC etc.

Small and large module re-use

Hierarchical organization of designs, EDA, et al.

....................
Embedded System Design Crisis

the 2\textsuperscript{nd} design crisis
“Pollack’s Law” (simplified)

- Growth factor
- Area efficiency
- Performance

μm
• Predictions require some history [Gordon Bell]

• Organizations usually behave poorer than anyone can predict [Gordon Bell]

• This especially holds for research funding and politics in Europe [R. H.]
The European Paradigm Shift Paradox

• When major funding agencies and their prominent advisors exclude an area, it becomes mainstream (e.g. elsewhere)

• the EU commission decided to exclude HDLs from funding* [CAVE workshop, Nice, France, 1989]

*) after having spent about 100 Mio ECU 1983-1989

• Similar errors:
  
  Ken Olson, 1977: "There is no reason anyone would want a computer in their home"

  Heinz Nixdorf: "We won't switch from Mercedes to Bicycle"

• the EU commission rejected all consortia applications on funds for research in Reconfigurable Computing [2003]
Reconfigurable Computing: a second programming domain

Migration of programming to the structural domain

The structural domain has become RAM-based

The opportunity to introduce the structural domain to programmers ...

... to bridge the gap by clever abstraction mechanisms using a simple new machine paradigm
**Datapath width**

1 bit CLB: fine grain

- **Word level CFB:** coarse grain

- **bundling of nibble or byte width CFBs:** multiple granularity
Ubiquitous Embedded Systems

Embedded System Engineering (ESE) requires:

- **HW / CW / ESW co-design onto highly programmable platforms (SoC)**

ESW and CW become main vehicle to product differentiation

ESE and CW become the main focus in system design

(Performance and) **Flexibility** are key issues

© 2004, reiner@hartenstein.de
Stanford Streaming Languages: DSP-C

set of language extensions to ISO C programming language allows application programmers describe key features of DSPs that enable efficient source code compilation:

- fixed point data types,
- divided memory spaces,
- circular arrays and pointers

DSP-C uses arrays

Sections of the arrays are selected for use in calculations using array indices or array range specifications

www.dsp-c.org
Brook defines much more abstract streams

Dynamic length streams

Could be multidimensional (but only fixed length??)

Entire stream is a monolithic object

Stream programming language for modern graphics hardware
"After a few years of looking at the problem, I realized that most of the application space could be described well with a stream-oriented communicating sequential processes model" Gokhale said, the compiler makes FPGA design available to software engineers, but they should still have an "abstract notion" of hardware: "One way to get performance is to tile application-specific arithmetic units across a chip," she noted. "Telling the compiler to unroll inner loops is a way to do that."

What's not needed, she said, is a knowledge of the hardware at a clock cycle level.

In contrast to SystemC, which provides both behavioral and structural views, Streams-C is purely behavioral. It assigns operations to clock cycles, thus providing behavioral synthesis.
Streams-C defines 1-D, array-like streams

- Arbitrary sections of the base arrays can be selected in advance for streaming into execution kernels

LANL open-source C compiler for reconfigurable logic, called Streams-C, accepts a subset of the C programming language, performs behavioral synthesis, and outputs synthesizable RTL VHDL code. It currently targets Xilinx Virtex-2000 devices on Annapolis Microsystems' Firebird board.

"After a few years of looking at the problem, I realized that most of the application space could be described well with a stream-oriented communicating sequential processes model" Gokhale said the compiler makes FPGA design available to software engineers, but they should still have an "abstract notion" of hardware: "One way to get performance is to tile application-specific arithmetic units across a chip," she noted. "Telling the compiler to unroll inner loops is a way to do that."

What's not needed, she said, is a knowledge of the hardware at a clock cycle level.

In contrast to SystemC, which provides both behavioral and structural views, Streams-C is purely behavioral. It assigns operations to clock cycles, thus providing behavioral synthesis.
Streaming Languages: Parallelism

- Independent thread parallelism
  - Stick with pthreads or other high-level definition
- Loop iteration, data-division parallelism
  - Divide up loop iterations among functional units
  - Loop iterations must be data-independent (no critical dependencies)
- Pipelining of segments of “serial” code
  - Find places to overlap non-dependent portions of serial code
    - Ex. 1: Start a later loop before earlier one finishes
    - Ex. 2: Start different functions on different processors
  - Harder than loop iteration parallelism because of load balancing
- Pipelining between timesteps
  - Run multiple timesteps in parallel, using a pipeline
  - Doesn’t necessarily require finding overlap of loops or functions — running them on different timesteps makes them data parallel
  - StreaMIT is best example of a language designed to optimize for this, and up to now I don’t think any of our proposals have addressed it
Semiconductor Revolutions

Makimoto's Wave

- Standard
- Custom

1957: TTL
1967: LSI, MSI
1977: µproc., memory
1987: ASICs, accel's
1997: Reconfigurable
2007:

vN machine paradigm

anti machine paradigm
How's next Wave?

Tredennick’s Paradigm Shifts

- Standard: hardwired
- Algorithm: fixed
- Resources: fixed

- Procedural Programming
- Algorithm: variable
- Resources: fixed

- Structural Programming
- Algorithm: variable
- Resources: variable

FPGAs

4th Wave?

Coarse grain RAs

Hartenstein’s Curve

no further wave!

© 2004, reiner@hartenstein.de

http://hartenstein.de
Mainstream Markets

- 1957: Mainframes
  - IBM

- 1967: Technology
  - TTL

- 1977: Technology and business model
  - LSI, MSI

- 1987: Technology
  - PIs, accel's

- 1997: Business model
  - PC, Intel

- 2007: Mainframes
  - Trittbrettfahrer

- Data streams ...

- Morphware

- Here?
Mathematics is a process of invention rather than of discovery; in that sense, it resembles art and music more than pure science.

John Horgan: „The End of Science Revisited“
SNN filter on KressArray

array size: 10 x 16 = 160 rDPUs

"Structured Configware Design" [R.H.]

by the way: example of scalability / relocatability by EDA support
The Digital Divide of Parallelism


The White House, Sept 2000: Clinton condemns Digital Divide in America

Digital divide, a phrase frequently mentioned across the media, refers to the differences between the "haves" and "have nots"

... distinguish between Parallel Processing scenes: have only instruction-stream-driven parallelism and have not other levels of parallelism
Migration of programming to the structural domain

The structural domain has become RAM-based

The opportunity to introduce the structural domain to programmers ...

... to bridge the gap by clever abstraction mechanisms using a simple new machine paradigm
Migration of programming to the structural domain

The structural domain has become RAM-based

Currently running: the next fundamental revolution after introduction of the microprocessor

However, CS curricula ignore this impact of Reconfigurable Computing – key issue in embedded systems ...

... causing the coming disaster by unqualified CS graduates pushing up the unemployment rate?
Massively Reducing Electricity Consumption and Greenhouse Gases by Innovative Computing Education

Reiner Hartenstein
Successful Proposal?

Teaching to Energy and Climate Experts the massive impact of Reconfigurable Computing?

10 minutes are too short!

Please, e-mail to reiner@hartenstein.de
The impact of Computers

Computers offer a much higher Potential than most other Energy and Climate Policy issues

The Climate Science Scenes are not aware, that the world-wide Energy Consumption by computer operations is higher than expected

Aggressive controversial discussions jeopardize realistic estimations
Computers everywhere
Our Computing Ecosystem

Visible and hidden Computers everywhere:
In PCs, laptops, peripherals in offices, homes & elsewhere
In entertainment equipment at home and elsewhere
In data centers, internet server farms, supercomputers
In base stations of wireless communication networks
In machines in industry, commercial buildings, homes etc.
In all kinds of vehicles, airplanes, trains, ships and more
In all kinds of portable equipment and more
This list is far from being complete
Computer Ecosystems: just one example
More Energy Guzzlers: Supercomputers
Google causes 2% of world-wide electricity consumption (> 1 million servers)


25% of Amsterdam's electricity goes into server farms

Server Farms
banks of Columbia river**

8 racks
water-cooled

Sun Microsystems' MD S20

75 MW

187.5 kW

8 racks

truck

each 6500 m²

10 American football fields

1) power for 40 000 homes (8 x Quincy)

Google causes 2% of world-wide electricity consumption (> 1 million servers)

Server Farms
banks of Columbia river**

8 racks
water-cooled

Sun Microsystems' MD S20

75 MW

187.5 kW

8 racks

truck

each 6500 m²

10 American football fields

1) power for 40 000 homes (8 x Quincy)

Google causes 2% of world-wide electricity consumption (> 1 million servers)
The von Neumann Syndrome

The predominance of tremendously inefficient von Neumann computers causes an unbelievably massive waist of energy

[2006 coined by Prof. C.V. “RAM” Ramamoorthy, UC Berkeley]

please, google „von Neumann Syndrome“

often software code sizes of astronomic dimensions
CPU: the tail wagging the dog

CPU: "Central Processing Unit"

However, it needs accelerators

"Central": it controls (almost) everything

Accelerators:
- Hardwired accelerators
- Reconfigurable accelerators
Reconfigurable Computing (RC)

reconfigurable accelerators

... are programmable

but different programming style:
non-von-Neumann

old stuff – but ignored by mainstream CS education and programmer training

The CS Education Dilemma

FPGA

Field- Programmable Gate Array

important acronym: are you familiar?

product introduced 1984
Pervasiveness of FPGAs & RC*

**Keyword** [March 8, 2009]

- FPGA
- Reconfigurable Computing

**other:**  
http://www.fpl.uni-kl.de/RCeducation07/pervasiveness.html

**international annual conference series on RC:**
- on RC directly: 42
- RC included: hundreds

*) Reconfigurable Computing

**Revenue for FY 2008:**
- Xilinx $ 1.84 billion
- Altera $ 1.2 billion

**world market:** ~ $ 5 billions

**Google**  
9,240,000  
141,000

**Yahoo**  
29,900,000  
635,000

- e.g. „FPGA & power“
**Most effective so far ...**

<table>
<thead>
<tr>
<th>Platform (compared to Beowulf cluster)</th>
<th>Speed-up factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI Altix 4700 with RC 100 RASC</td>
<td><strong>28514</strong></td>
</tr>
</tbody>
</table>

**DES breaking** [T. Elghazawi et al.: IEEE COMPUTER, Febr. 2008]

![FPGA](image_url)
Energy Saving factor: ~10% of speedup factor
Most effective so far ...

<table>
<thead>
<tr>
<th>Platform (compared to Beowulf cluster)</th>
<th>Speed-up factor</th>
<th>Power saving factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI Altix 4700 with RC 100 RASC</td>
<td>28514</td>
<td>3439</td>
</tr>
</tbody>
</table>

**DES breaking** [T. Elghazawi et al.: IEEE COMPUTER, Febr. 2008]

FPGA

\[ \sim 12\% \]
Drastically less Equipment needed

For instance: a hangar full of racks replaced by a single rack & without air conditioning.
We need a Paradigm Shift

Obsolete: CPU

Not the new model: reconfigurable accelerators

But this is the new model: CPU

New model: FPGA

Platform FPGAs include CPUs
We need Dual-Rail Education

New model:
FPGA
Platform FPGAs include CPUs

CPU
reconfigurable accelerators

Rocket
IO

Power PC
Core

On Chip
Memory Controller

Embedded
RAM

Courtesy XILINX Corp.
The Programmer Education Dilemma

We need a world-wide changeover of many applications to FPGAs

We‘ve yet to train and qualify a sufficiently large population of programmers

Advanced training needs support everywhere: at regional, national, and at global level

Software / Configware  Co-Education needed
Dual-Rail Programming Education

CPU
Central Processing Unit

FPGA
Field-Programmable Gate Array
We need „une’ Levée en Masses“
New Horizons of Energy Policy

To cope with rising cost and declining availability of energy resources we urgently need a paradigm shift in computing.

The solution is Reconfigurable Computing - FPGA.
Convincing Proposal?

Teaching to Energy and Climate Experts the massive impact of Reconfigurable Computing?

10 minutes are too short!

Please, e-mail to reiner@hartenstein.de
thank you for your patience
END
backup for discussion
The von Neumann Syndrome

the tremendous inefficiency of computers causes immense electricity consumption

More power for creating foam than to accelerate the vessel?
<table>
<thead>
<tr>
<th>Method</th>
<th>Potential: Slashing* down to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building and infrastructure solution</td>
<td>(side effect)</td>
</tr>
<tr>
<td>Computer architecture solution</td>
<td>5%</td>
</tr>
<tr>
<td>Microchip circuit &amp; technology solution</td>
<td>50%</td>
</tr>
<tr>
<td>Programming methodology solution</td>
<td>5%</td>
</tr>
<tr>
<td>Green Computing</td>
<td>40%</td>
</tr>
</tbody>
</table>

*) estimated
The „von Neumann“ mainframe

The contemporary basic mind set of programmers is still tape-oriented.

introduced early 40ies

Time domain: instruction streams, controlled by program counter

notorious headache w. parallelism

http://hartenstein.de © 2009, reiner@hartenstein.de
## von Neumann style vs. Reconfigurable Computing

<table>
<thead>
<tr>
<th>Machine Paradigm</th>
<th>CPU</th>
<th>Reconf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>instruction streams</td>
<td>data streams</td>
</tr>
<tr>
<td>Controlled by</td>
<td>program counter</td>
<td>data counters</td>
</tr>
<tr>
<td>Program Source</td>
<td>software</td>
<td>configware &amp; flowware</td>
</tr>
</tbody>
</table>

**CPU**

- Von Neumann Computing
- Reconfigurable Computing

**Operations**
- instruction streams
- data streams

**Controlled by**
- program counter
- data counters

**Program Source**
- software
- configware & flowware
# Programming in Time vs. Programming in Space

<table>
<thead>
<tr>
<th>machine paradigm</th>
<th>CPU</th>
<th>reconf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>time domain</td>
<td>Von Neumann Computing</td>
<td>Reconfigurable Computing</td>
</tr>
<tr>
<td>instruction streams</td>
<td>data streams</td>
<td></td>
</tr>
<tr>
<td>software</td>
<td>flowware</td>
<td></td>
</tr>
<tr>
<td>program counter</td>
<td>data counters</td>
<td></td>
</tr>
</tbody>
</table>

(space domain) (hardwired) configware
Flash Strawman Example

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clock</th>
<th>Peak/ Core (Gflops)</th>
<th>Cores/ Socket</th>
<th>Sockets</th>
<th>Cores</th>
<th>Power</th>
<th>Cost 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Opteron</td>
<td>2.8GHz</td>
<td>5.6</td>
<td>2</td>
<td>890K</td>
<td>1.7M</td>
<td>179 MW</td>
<td>$1B+</td>
</tr>
<tr>
<td>IBM BG/P</td>
<td>850MHz</td>
<td>3.4</td>
<td>4</td>
<td>740K</td>
<td>3.0M</td>
<td>20 MW</td>
<td>$1B+</td>
</tr>
<tr>
<td>Green Flash / Tensilica XTensa</td>
<td>650MHz</td>
<td>2.7</td>
<td>32</td>
<td>120K</td>
<td>4.0M</td>
<td>3 MW</td>
<td>$75M</td>
</tr>
</tbody>
</table>

[Courtesy Horst Simon et al., LBNL, Berkeley]

Energy saving factor: 60
Green Computing – Green IT Technology

200 watts
76" HDTV

550 watts
Plasma TV

380 watts
video game
with IBM
Cell BE

low power
circuit design
(we also need
conservative
approaches)
Conclusions

A von-Neumann-only strategy can never be the solution.

We need a massive Software to Configware Migration.

Established technologies are available and we can still use standard software and their tools.

**Configware skills and basic hardware knowledge are essential qualifications for programmers.**

**We urgently need a fundamental CS Education and Research Revolution for dual-rail-thinking.**
Currently the scenario is comparable to the VLSI design crisis around 1980.

Missing reply to the End of Moore’s Law

The best time to begin is right now

The likeliness of success stems from the urgent need to cope with a massive threat

We need „une' Levée en Masses“
Mouse Brain Simulation

- E. g. Harvard’s Center for Brain Science.
- solving an astronomical number of coupled partial differential equations.
- A digital computer capable of coping with this staggering workload would need to be the size of a small city, and
- it would require several dedicated nuclear power plants.
- the human brain has 100 billion neurons, this represents 1/10,000 of a human brain: $1,000,000,000,000$
VLSI design education spreading rapidly
1980 - 1983
world-wide

Carver Mead
Lynn Conway

new text book
1980
Teaching to energy and climate experts the impact of reconfigurable computing?

10 minutes are too short!